CMR Bulk Update

Tables

2 tables for tracking the status of a bulk update task:

Bulk update task table - overall status

- Task Id
- Provider Id
- Request JSON body
- Status
- Status message

Bulk update status table - individual collection statuses

- Task Id
- Concept Id
- Status
- Status message

Ingest Endpoint

POST ingest/providers/<provider-id>/bulk-update/collections

POST JSON:

- List of collections to update - either concept id or short-name/version *(short-name/version currently not supported)*
- Field to update - required
- Type of update to make - required
- Update value
- Find value

Field to update and type of update are enumerations.

The update and find values are the UMM object or a subset of the UMM object. For example, the science keywords update value would be the full science keyword {"Category": "Cat1"...} and the find value would take all or part of a science keyword and match on the given fields.

The ingest endpoint:

- Validates the parameters against a JSON schema and rules i.e. the new value and find value can be required based on type of update.
- Checks ACL update permissions for provider
- If short-name/version supplied, get concept ids *(on hold for now - will be later functionality)*
- Writes to the task status and collection status tables in a transaction and gets the task id
- Queues a bulk update message
- Returns status code and task Id if no error, otherwise error messages

Validations

Bulk update POST body validations:

- Short-name/version Ids required or concept-ids required, min 1 item in the list
- Type of change required
- Field to update required
- Field to update valid (in map of fields)
- If NOT type CLEAR_FIELD or FIND_AND_REMOVE, New value required
- If type FIND_AND_REMOVE or FIND_AND_REPLACE, Find value required

Sample POST JSON

```
{
    "concept-ids": [
        "C1",
        "C2",
        "C3"
    ],
    "update-type": "FIND_AND_REPLACE",
    "update-field": "SCIENCE_KEYWORDS",
    "find-value": {
        "Category": "EARTH SCIENCE",
        "Topic": "HUMAN DIMENSIONS"
    },
    "update-value": {
        "Category": "EARTH SCIENCE",
        "Topic": "HUMAN DIMENSIONS",
        "Term": "ENVIRONMENTAL IMPACTS",
        "VariableLevel1": "HEAVY METALS CONCENTRATION"
    }
}
```

Status Endpoints

Both require ACL read permissions for provider.

Endpoint to get all task statuses by provider: GET ingest/providers/<provider-id>/bulk-update/collections/status

Returns list of:

- Task Id
- Task Status enum ("IN_PROGRESS", "COMPLETE")

For each task for that provider.

Endpoint to get individual task status: GET ingest/providers/<provider-id>/bulk-update/collections/status/<task-id>

Returns:

- Task Status
- Task Status Message
- For each failed collection:
    - Concept Id
    - Collection update status message

Bulk Update Queue

Processing a bulk update message:

- For each collection, publish individual bulk update messages

Processing a message on the queue:

- Check if the collection status is cancelled. If so, skip. *(This will not be implemented in the first iteration)*
- Pull the collection from metadata DB
- Translate collection to UMM
- Make field update change
- Perform UMM validations: schema, business rules
- Translate back to native format
- Perform XML schema validation
- Save to metadata db with revision Id
    - If concurrency failure, fail the message and will be retried
- Re-index collection

- In a transaction
    - Update the bulk update status table for the collection
    - Check to see if the overall bulk update is done and update the bulk update task table

AWS

Use the above design with an SNS/SQS message queue. For each collection, we write a message to the queue to process that collection (may want to do this in batches).

We have a lambda function that does the collection message processing outlined above. The lambda function would live in a separate project that refers to umm-spec-lib to do the updates. Use transmit lib to talk to metadata db.

Considerations:

- Lambda start-up time
- Lambda limitations in terms of memory, disk space, and execution time.
- Running too many concurrent lambdas

Testing

- Unit test update function in umm-spec-lib
- Test to make sure states are tracked and updated correctly
- Be able to run everything locally/in-memory

Cancel Operation

Add an endpoint to cancel a bulk update by task id. Need to check ACL permissions. Mark all collection entries in the bulk update status table as cancelled if they have not already been processed. Update the bulk update task table with the overall status: cancelled or partial cancel.

DB Cleanup

Add a process that goes through and cleans up old bulk update db status rows periodically.

Current Assumptions

- Starting off only updating certain fields - GCMD keyword fields
- Cancel not part of initial implementation
- If multiple bulk update processes running updating the same field on the same collection, no guarantee that you'll get the latest update